AMENDMENTS TO THE SPECIFICATION

[0001] **This application is a continuation of U.S. Serial Number 09/280,112 that was filed on March 26, 1999 under the same title and to the same inventors as the present application.** This application is related to the following applications: U.S. **Patent Nos. 6,055,211 [Application Serial Nos. 08/887,876]** for "FORCE PAGE **[ZERPO] ZERO [PAGEING] PAGING** SCHEME FOR MICROCONTROLLERS USING DATA ACCESS MEMORY" **[on 07/03/98]** by Randy L. Yach, et al. **[(MTI-1225)]**; **5,905,880 [08/937,682]** for "ROBUST MULTIPLE WORK INSTRUCTION AND METHOD THEREFOR" **[on 08/29/98]** by Rodney J. Drake, et al. **[(MTI-1254)]**; **6,192,463 [08/946,426]** for "PROCESSOR ARCHITECTURE SCHEME **WHICH USES VIRTUAL ADDRESS REGISTERS TO IMPLEMENT DIFFERENT ADDRESSING MODES [FOR IMPLEMENTING VARIOUS ADDRESSING MODES]** AND METHOD THEREFOR" **[on 10/07/97]** by Sumit Mitra, et al. **[(MTI-1265)]**; **6,243,798 [08/958,940]** for "**COMPUTER [A]** SYSTEM FOR ALLOWING A TWO WORD INSTRUCTION TO BE EXECUTED IN **THE SAME NUMBER OF CYCLES AS A SINGLE WORD JUMP INSTRUCTION [A SINGLE CYCLE AND METHOD THEREFOR]**" **[on 10/28/98]** by Rodney J. Drake, et al. **[(MTI-1298)]**; **6,029,241 [08/959,405]** entitled "PROCESSOR ARCHITECTURE SCHEME HAVING MULTIPLE **BANK ADDRESS OVERRIDE SOURCES FOR SUPPLYING [SOURCE FOR SUPPLYING BANK]** ADDRESS VALUES AND METHOD THEREFORE" **[filed on 10/28/97]** by Igor Wojewoda, Sumit Mitra, and Rodney J. Drake **[(MTI-1299)]**; **6,098,160 [08/959,559]** for "DATA POINTER FOR OUTPUTTING INDIRECT ADDRESSING MODE ADDRESSES WITHIN A

[SIGNLE] **SINGLE** CYCLE AND METHOD THEREFOR" **[on 10/29/98]** by Rodney J. Drake, et al. **[(MTI-1300)]**; **5,958,039** **[08/958,939]** for "**MASTER-SLAVE LATCHES AND** **[PRE-DECODED STACK POINTER WITH]** POST INCREMENT/DECREMENT OPERATION" **[on 10/28/98]** by Allen, et al. **[(MTI-1306)]**; and **5,987,583** **[08/959,942]** for "PROCESSOR ARCHITECTURE SCHEME AND INSTRUCTION SET FOR MAXIMIZING **[AVILABLE]** **AVAILABLE** OPCODES AND **[FOR IMPLEMENTING VARIOUS]** ADDRESSING **SELECTION** MODES" **[on 10/29/97]** by Triece, et al. **[(MTI-1314)]** which are hereby incorporated herein by reference for all purposes.

[0004] As shown in Figure **[1]** **2**, the Harvard architecture has the program memory 26 and data memory 22 as separate memories and are accessed by the CPU 24 from separate buses. This improves bandwidth over traditional von Neumann architecture (shown in Figure 3) in which program and data are fetched by the CPU 34 from the same memory 36 using the same bus. To execute an instruction, a von Neumann machine must make one or more (generally more) accesses across the 8-bit bus to fetch the instruction. Then data may need to be fetched, operated on, and possibly written. As can be seen from this description, that bus can **[be]** **become** extremely **congested** **[conjested]**.

[0014] An "Instruction Cycle" consists of four Q cycles (Q1, Q2, Q3, and Q4) as shown in Figures 4 that comprise the **[Tcy]** **TCY** as shown in Figures 4 and 5. Note that in Figure 5, all instructions are performed in a single cycle, except for any program

branches. Program branches take two cycles because the fetch instruction is "flushed" from the pipeline while the new instruction is being fetched and then executed.

[0016] While the prior art microcontrollers were useful, the various modules could not be emulated. Moreover, the type of microcontroller as described in Figure 1 could not linearize the address space. Finally, the prior art microcontrollers are susceptible to compiler-error problems. What is needed is an apparatus, method, and system for a microcontroller that is capable of **[liniarizing]** **linearizing** the address space in order to enable modular emulation. There is also a need in the art for reducing compiler errors.

[0018] The present invention can directly or indirectly address its register files or data memory. All special function registers, including the Program Counter (PC) and Working Register (W), are mapped in the data memory. The present invention has an orthogonal (symmetrical) instruction set that makes it possible to carry out any operation on any register using any addressing mode. This symmetrical nature and lack of 'special optimal situations' make programming with the present invention simple yet efficient. In addition, the learning curve for writing software applications is reduced significantly. One of the present invention's enhancements over the prior art allows two file registers to be used in some two operand instructions. This allows data to be moved directly between two registers without going through the W register**; and thus** **[. Thus]** increasing performance and decreasing program memory usage.

[0138] A selection circuit 108**[is]** **is** coupled to the data memory 104 through an address latch 102. The selection circuit 108 is used for selecting one of the plurality of sources that supply the bank address values in the data memory 104.

[0150] bit 3 is the "OV" Overflow bit. This bit is used for signed arithmetic (2's complement). It indicates an overflow of the 7-bit magnitude, which causes the sign bit (**bit 7** **[bit7]**) to change state. For this bit, 1 = Overflow occurred for signed arithmetic, (in this arithmetic operation), and 0 = No overflow occurred.

[0157] The Program Counter (PC) 168 is up to a 21-bit register as shown in Figure 16. PCL 184, the low byte of the PC 168, is mapped in the data memory 104 (see Figure 6). PCL 184 is readable and **writeable** **[writable]** just as is any other register. PCH 182 and PCU 180 are the high bytes of the PC and are not directly addressable. Since PCH 182 and PCU 184 are not mapped in data or program memory 160, registers PCLATH 178 (PC high latch) and PCLATU 176 (PC upper latch) are used as holding latches for the high bytes of the PC 168.

[0171] Since the Top-of-stack (TOS) is readable and **writeable** **[writable]**, the ability to push values onto the stack and pull values off the stack without disturbing normal program execution is a desirable option. To push the current PC value onto the stack, a PUSH instruction can be executed. This will push the current PC value onto the stack; setting the TOS = PC and PC = PC + 2. The ability to pull the TOS value off of the stack and replace it with the value that was previously pushed onto the stack, without

disturbing normal execution, is achieved by using the POP instruction. The POP instruction pulls the TOS value off the stack, but this value is not written to the PC; the previous value pushed onto the stack then becomes the TOS value.

[0177] The preferred embodiment of the present invention has up to a **2 Megabyte [2Megabyte]** (2M) x 8 user program memory space. The program memory space is primarily to contain instructions for execution, however, data tables may be stored and accessed using the table read and write instructions. Another 2M x 8 test program memory space is available for test ROM, configuration bits, and identification words.

[0187] The extended microcontroller mode and microprocessor modes are available only on devices which have the external memory bus defined as part of the I/O pins. Table 2 lists which modes can access internal and external memory. Figure 33 illustrates the device memory map in the different program modes.

Table 2 Device Mode Memory Access

Operating Mode	Internal Program Memory	External Program Memory
Microprocessor	No Access	Execution / TBLRD / TBLWT
Extended Microcontroller	Execution / TBLRD / TBLWT	Execution / TBLRD / TBLWT
Protected Extended Microcontroller	Execution	Execution / TBLRD / TBLWT
Microcontroller	Execution / TBLRD / TBLWT	No Access
Protected **Microcontroller [Micrcontroller]**	Execution / TBLRD	No Access

[0197] The WAIT_<1:0> bits in the MEMCON register will select 0,1,2 or 3 extra **[Tcy]** **TCY** cycles per memory fetch cycle. The wait cycles will be effective for table reads and writes on a 16-bit interface. On an 8-bit interface, for table reads and writes, the wait will only occur on the Q4.

[0198] The default setting of the wait on power up is to assert a wait of the maximum of the **[3Tcy]** **3 TCY** cycles. This insures that slow memories will work in microprocessor mode immediately after reset. A configuration bit, called WAIT, will enable or disable the wait states. Figure 39 illustrates the 16-bit interface and Figure 40 illustrates the 8-bit, in both cases showing program memory instruction fetches with no waits and table reads with wait states.

[0220] Each INDF register has four addresses associated with it. When a data access is done to the one of the four INDF locations, the address selected will configure the FSR register to **[(see also Section 3.14.6.3 for indirect addressing with offset)]**: